ICAS 2016
30th Congress of the International Council of the Aeronautical Sciences
DCC, Daejeon, Korea ; September 25-30, 2016

REAL-TIME GENERATION OF WAYPOINTS ENHANCING UAV SURVIVABILITY

Sanghyuk Park*, Kuk-Kwon Park, Sang-Sup Park**, and Chang-Kyung Ryoo****
***LIG Nex1, ****Inha University**

Keywords: *Path planning, Survivability, Voronoi diagram, VVCO*

Abstract

This paper suggests the path planning algorithm with optimized waypoints, which maximizes the survivability of a UAV from the hostile environment. The algorithm optimizes the waypoints with several steps. First, initial waypoints and path in this algorithm are obtained by Voronoi diagram and Dijkstra algorithm, and then each waypoint is optimized by repeating local optimization. The concept of optimization is that only one waypoint is optimized under the assumption that nearby waypoints are fixed and repeats this procedure for whole waypoints. In this method, the waypoints are obtained in the sense of maximizing survivability in real-time. A simulation is performed to illustrate this algorithm.

1 Introduction

The missions of UAV(unmanned aerial vehicles), such as cruise missiles, unmanned reconnaissance aircraft, and unmanned combat aerial vehicles, include performing strategic reconnaissance or hitting target objects after flying over enemy territory in hostile environments from a safe area far away from the battlefield. Therefore, for the operation of UAVs, it is essential to determine the waypoints to successfully navigate through from the start to the goal point. Such waypoints should be set to define the navigation route beforehand so that a UAV can reach its target and accomplish the mission most efficiently.

A large number of studies have been conducted to explore waypoint determination techniques, not only for path planning for UAVs, but also for robot navigation. Methods for waypoint initialization can be largely divided into two categories: direct selection of waypoints and additional selection of waypoints on a planned route. Voronoi diagrams [1, 2, 3] and visibility graphs [4, 5] belong to the first category; potential field [6, 7, 8, 9], trajectory optimization [10, 11, 12, 13], and sampling-based planning [14, 15] techniques are some examples of the second category. Of these methods, the Voronoi diagram enables the construction of simple and reliable route waypoint calculation algorithms capable of rapid calculation of the fastest route waypoints and easy integration of spatial threatening elements. Voronoi diagram-based algorithms yield only linear roadmaps and thus require corrections in path planning for the dynamic characteristics of flying vehicles. However, path planning for high-velocity and long-distance navigation of cruise missiles and high-altitude unmanned reconnaissance aircraft generally involves straight-line segments, and because the flying vehicles are preprogramed or remotely controlled to follow these linear routes, such linear path planning algorithms may prove more useful.

In this study, a Voronoi diagram method was employed for waypoint initialization in path planning for a UAV penetrating enemy territory. For problem definition, it was assumed that each threat has the shape of an exponential function. The waypoint determination problem under different-size threats has been addressed in many studies. Representative of such studies includes the circle set Voronoi diagram [16], in which circles of different radii are presented as threats, and the distances between threat boundaries are bisected; the multiplicatively weighted Voronoi diagram [17] for equidistant divisions using arbitrary spatial distances

generated by weighting the Euclidean distances between threats; and the improved Voronoi diagram [18], in which an internal dividing point between two waypoints does not bisect the distance between them by giving them different weights. However, such methods are unsuitable for linear path planning, given that they involve curved paths, or are not readily applicable to problems involving distance-dependent weights.
In this study, in an attempt to overcome such problems, waypoints were initialized first using a Voronoi diagram and then optimized in order to avoid the threats having different sizes and shapes. Additionally, a method for sequential optimization of the positions of initially computed waypoints was applied to solve the problem of the computation time required for implementing the optimization on the entire set of waypoints.

2 Algorithm Design and Evaluation

2.1 Path Planning

This paper aims to design a path planning algorithm that maximizes the survivability of a UAV from a safe start point to the goal point, maneuvering through the threats encountered in enemy territory. Assumptions were made that the position of each threat is known, as is the survivability from each threat, and that the planned path comprises straight lines.
The formulas below represent the monotone increasing survivability function, where x is the distance between the threat and the path and $f(x)$ represents survivability.

$$\Delta f(x) = f(x') - f(x) > 0 \quad for \;\; \Delta x = x' - x > 0, 0 \le f(x) \le 1 \qquad (1)$$

The smallest value in the same straight-line segment is taken as the survivability from each threat, as represented by the kill probability in Fig. 1.
To implement the optimization of a Voronoi diagram, the objective function should be defined. The optimization problem was defined as the calculation of the performance evaluation coefficient expressed by (2) using the survival

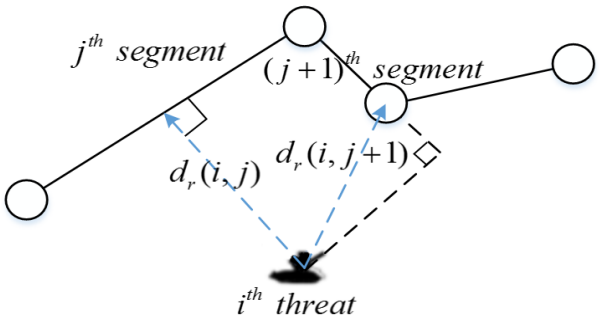

Fig. 1. Kill probability of each segment.

probability of each straight-line segment and maximization of the performance evaluation coefficient:

$$S = \prod_{j=1}^{M} \max_{i=1,\ldots,N} (S_{i,j}) \qquad (2)$$

where $S_{i,j}$ is survivability of the i-th flight segment of a UAV against the j-th threat, N and M denote the numbers of threats and straight-line segments, respectively.

2.2 Development of a Path Planning Algorithm

The waypoint optimization for path planning should be preceded by the determination of the number of waypoints and selection of the initial positions. Therefore the path-planning algorithm developed in this study has a two-step structure; optimal number of waypoints and their positions are determined first, which facilitates the solution and reduces the time requirement for computation in the second step.
Waypoints can be initialized using the grid-based Voronoi diagram employed in path planning studies, and the number of waypoints can be determined using an algorithm for finding shortest paths; Dijkstra's algorithm. If these algorithms are applied under the same combat environments, waypoint candidates and

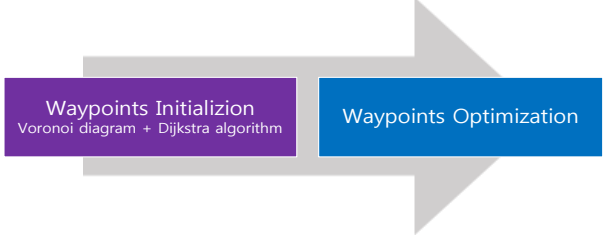

Fig. 2. Path planning algorithm consisting of two steps.

combat environments can be accurately matched, and by using the same environmental parameters and weights, the same waypoint set can be obtained at each iteration. Thus, a reliable algorithm can be constructed that is capable of deriving stable solutions. In the second step of optimization, the initial waypoints are corrected, resulting in various flight routes depending on the price function and constraint condition definitions.

2.3 Waypoint Initialization

2.3.1 Voronoi Diagram

A Voronoi diagram is composed of lines connecting the equidistant points between two nearest points or object boundaries in a set of points or objects in a Euclidean space. A Voronoi diagram is a free-space data structure in which the free space between points plays an important role because a plane containing objects is partitioned into polygonal Voronoi regions. A Voronoi region is defined as the region closer to a given point than to any other points in space, and the boundary of each region that is equidistant from the two closest points is termed a Voronoi edge. Points within a Voronoi region can be assumed to be terrestrial objects or anti-air threat sites, and a Voronoi edge can be considered a flying path avoiding such threats, and is thus widely used for operations such as collision avoidance or path planning. The following is a list of definitions of the terms specific to a Voronoi diagram as illustrated in Fig. 3.

•Voronoi Cell : the reference point for forming a region

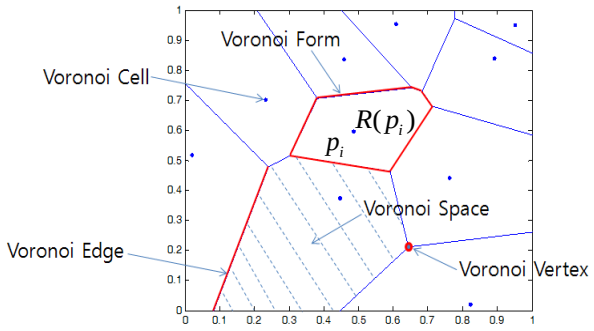

Fig. 3. Structure of a Voronoi diagram in the plane.

•Voronoi Site/Space : a region pertaining to each cell constrained by boundaries

•Voronoi Edge : an edge formed by two adjacent Voronoi spaces

•Voronoi Vertex : a vertex formed by three or more Voronoi spaces

•Voronoi Form : the shape taken by a Voronoi space

Given that an arbitrary d -dimensional space R^d contains n points $p_1, p_2, ..., p_n$, the Voronoi region for i -th point p_i, $R(p_i)$, is defined as

$$R(p_i) = \left\{ p \in R^d \mid \|p - p_i\| < \|p - p_j\| \right\} \quad \text{for } i \neq j \tag{3}$$

where $\|p - p_i\|$ denotes the Euclidean distance between point p and point p_i. If point p is closer to p_i than to p_j, point p belongs to the Voronoi region of p_i, and if they are equidistant, point p belongs to the Voronoi edge.

When applied to a flying vehicle navigating through a defense system such as radar or an anti-missile system, each threat has a different impact, and different magnitudes of impact can be expressed with circles of different sizes. Therefore, a Voronoi diagram for circles, and not for simple points, has a wider scope of application, and a large number of studies have explored this aspect. However, because of the constraint condition of path planning of a flying vehicle that the flight path is a series of straight lines, the curved edges of a circle Voronoi diagram make it unsuitable for UAV path planning. This problem was overcome in this study by generating a Voronoi diagram for unweighted points in step 1 and applying weights when performing the optimization of the initialized path in step 2.

2.3.2 Dijkstra's Algorithm

Although a Voronoi diagram generates a path that minimizes threats in consideration of the distribution of the threats, this does not necessarily conform to the flight path selection. In actual path planning, an algorithm is used for finding the shortest possible path from the start to the goal point following the Voronoi edges.

For this, we chose Dijkstra's algorithm, which is the most widely used path-finding algorithm. Dijkstra's algorithm is a label-setting algorithm most widely used because it is easy to understand and applicable to any problems for finding the shortest paths. From a given initial node, it selects the least-cost path from all alternative paths based on the principle of optimality and labels the corresponding node as a permanent node. This has the advantage of requiring the minimum necessary computation time, but the disadvantage of the unavailability of a detour path under the same environment. Moreover, the weights of all links should not have negative values, i.e., $w_1 + w_2 \geq w_1$, which means that the higher the number of links constituting a path, the longer the path [19].

2.3.3 Waypoints Selection

Fig. 5 illustrates an example of a path planning procedure as described above as step 1. First, a combat environment is modeled, then a Voronoi diagram is generated considering each threat, and finally, the initialized path is determined using Dijkstra's algorithm. This may be an optimal flight path if all threats have the same size and the flight path contains only straight lines. This approach has the advantages of deriving computationally stable solutions and short computation time. However, because the initialized path is no longer an optimal path if threats have different sizes, optimization should be performed in step 2 on the basis of the initialized path established in step 1.

2.4 Waypoint Optimization

In order to perform waypoint optimization on the basis of the waypoint set yielded by the Voronoi diagram, the optimization problem is defined as follows:

- Objective function

$$J = S = \prod_j \max_i (S_{i,j})$$

- Optimization parameters

$$\left[V_1(x_1, y_1), V_2(x_2, y_2), \cdots, V_n(x_n, y_n)\right]$$

- Constraint condition

$$x_{\min} \leq x_i \leq x_{\max} \qquad (i = 1, 2, ..., n)$$

$$y_{\min} \leq y_i \leq y_{\max}$$

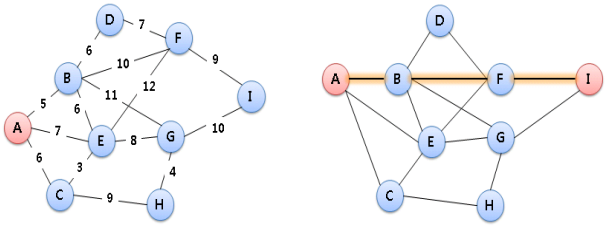

Fig. 4. Dijkstra's Algorithm [20].

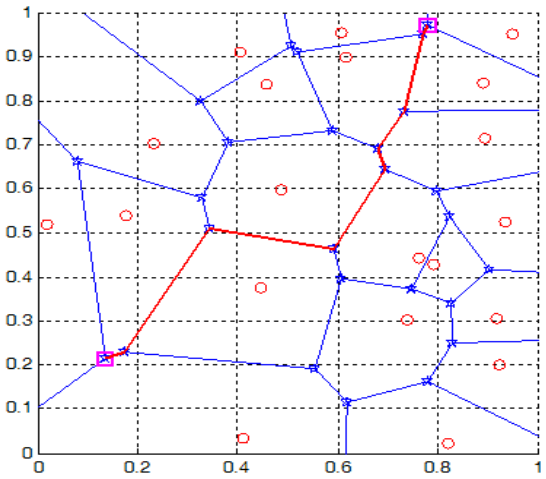

Fig. 5. Example of waypoint selection.

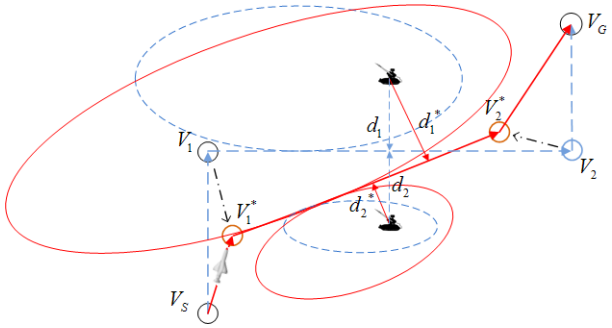

Fig. 6. Concept of waypoint optimization.

The optimization problem posed above is finding the waypoint positions minimizing the kill probability. In doing so, besides the kill probability, the total flight time is also considered in the objective function so that the entire flight path can be kept as short as possible, thereby applying a smaller weight to the flight time than to the kill probability. As depicted in Fig. 6, when two threats with different weights are assumed to be present, the initial waypoints V_1 and V_2 obtained from the Voronoi diagram are selected as equidistant points without regard to the different weights. Because of the difference in impact between the two threats,

however, the actual survivability is calculated to be lower. In other words, the perpendicular distance from the threat center to the flight path is identical for d_1 and d_2, but the survivability for $P_{surv}(d_1)$ is smaller than that for $P_{surv}(d_2)$. Therefore, d_1^* and d_2^* should be found so that the overall survivability P_{surv} can be maximized. Eventually, the survivability of two waypoints V_1 and V_2 can be enhanced when they are shifted to V_1^* and V_2^*.

This type of optimization problem can be problematic if the optimization technique is based on gradient, as is the case with the sequential quadratic programming (SQP). By the nature of optimization, it is logical that computation takes longer as the number of optimization parameters increases. The path planning in this paper involves an additional element that increases the computation time. The objective function is expressed as the product of survival probabilities. This implies that as the numbers of waypoints and threats increase, the overall survivability approximates zero, and optimization is impeded through the decrease in sensitivity to the overall survivability for waypoint positions. To solve this problem, this paper proposes a technique for sequentially optimizing specific waypoint.

2.4.1 Voronoi Vertex Circulation Optimization

Voronoi vertex curculation optimization (VVCO) begins by optimizing the first waypoint V_1 on the path selected through waypoint initialization, as illustrated in Fig. 7. For this technique, the positions of the previous (start) and next (V_2) points are fixed. Fig. 8 shows the waypoint V_1 is optimized and shifted to V_1^*, and V_2 is prepared to be optimized with V_1^*, and V_3 is fixed.

Likewise, Fig. 9 shows the waypoint V_2 is optimized and shifted to V_2^*, and V_3 is prepared to be optimized with V_2^* and goal point is fixed. Finally, Fig. 10 shows the state of completed optimization of all three waypoints between the start and goal points.

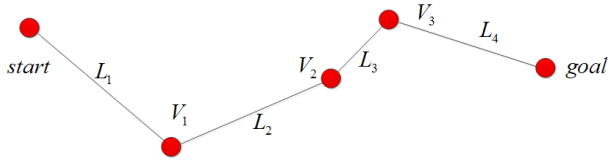

Fig. 7. Initial selection of waypoints.

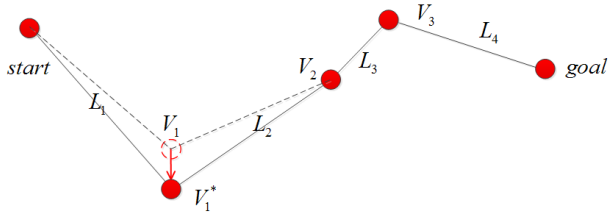

Fig. 8. Update of the 1st waypoint.

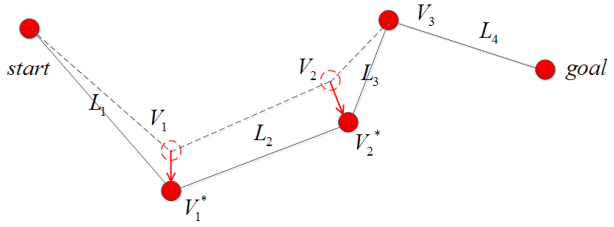

Fig. 9. Update of the 2nd waypoint.

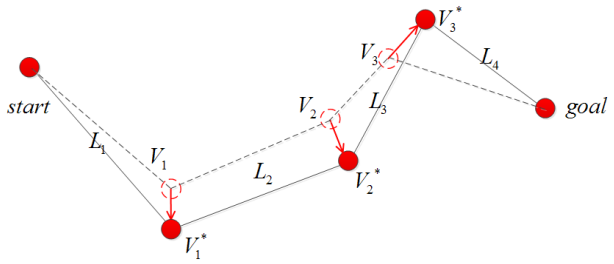

Fig. 10. Update of the 3rd waypoint.

The result of one round of individual optimization of the waypoints between the start and goal points, as shown in Fig. 7–10, is a sequential optimization performed to optimize V_1^* with respect to the start point and V_2, which may not be optimal for V_2^*. Therefore, the same procedure is applied many times until the iteration converges to the optimal overall survivability. In other words, if the survivability evaluation results for all waypoints fall under a certain threshold value in the preceding and present loops, then they are considered to be converged and the algorithm is terminated.

2.4.2 Optimality of Circulation Optimization

An important task is to evaluate the ability of the circulation optimization method explained previously, in which the waypoints between the fixed start and goal points are optimized one by

one, to yield the optimal solutions for all waypoints. As depicted in Fig. 7–10, each waypoint is optimized, with the neighboring waypoints fixed, whereby it is assumed that each waypoint is always provided with the optimal solution. First, let S_j be the survivability of the j-th path section from all threats ($n = N$) in the initialized path as shown in Fig. 7; the overall survivability of the entire path having M waypoints can be expressed by

$$S = S_1 \times S_2 \times \cdots \times S_{M+1} \tag{4}$$

The value $S_1^* \times S_2^*$ calculated from the new path after moving V_1 to V_1^* as a result of its optimization is inevitably larger than the pre-optimization value $S_1 \times S_2$. The same applies to all other optimization results ($S_2^{**} \times S_3^* > S_2^* \times S_3$ after moving V_2 to V_2^*, etc.). In terms of the changes in overall survivability, this can be expressed as

$$\begin{aligned} S &= S_1 \times S_2 \times S_3 \times \cdots \times S_M \times S_{M+1} \\ &\le S_1^* \times S_2^* \times S_3 \times \cdots \times S_M \times S_{M+1} \\ &\le S_1^* \times S_2^{**} \times S_3^* \times \cdots \times S_M \times S_{M+1} \\ &\vdots \\ &\le S_1^* \times S_2^{**} \times S_3^{**} \times \cdots \times S_M^* \times S_{M+1} \\ &\le S_1^* \times S_2^{**} \times S_3^{**} \times \cdots \times S_M^{**} \times S_{M+1}^* \\ &= S^* \end{aligned} \tag{5}$$

As shown in (5), S^*, which is the post-optimization survivability after each run of the loop from the first waypoint V_1 to the last waypoint V_M, is always larger than the pre-optimization survivability S. The exception may be the case in which the initial waypoint value is already the optimal value.

However, the initially optimized V_1^* is no longer the optimal solution after V_2 has moved to V_2^*. V_1^* is then re-optimized with the start point and V_2^* is fixed to obtain the optimal solution V_1^{**}, thus increasing the overall survivability. Consequently, the overall survivability S^{**} obtained after sequentially optimizing all waypoints becomes higher than S^* again, leading to a continuous increase in the overall survivability as the loop iteration continues. Thus,

$$S^{*(n-1)} \le S^{*(n)} \le 1 \tag{6}$$

Given that the survivability cannot ceaselessly increase, the increase rate will likely fall at a certain survivability value less than the maximum value 1. In our study, an upper limit value was set, and the loop was configured to discontinue when the survivability increase rate falls below this limit:

While $(S^{*(n)} - S^{*(n-1)} > \varepsilon)$

…

end

2.4.3 Voronoi Vertex Local Optimal Solution

Earlier, it was assumed that the optimal solution can be derived for each waypoint through optimization. If each threat can be represented by a monotone increasing survivability function, optimal solutions can be obtained using a gradient-based SQP under the condition that the combat environment presenting multiple threats has extreme values. In order to verify this theory, the following navigation space was assumed. Fig. 11 depicts the initialized path consisting of four waypoints between the start and goal points in a space of 30 km × 30 km with 10 threats of different sizes distributed along the path, using a Voronoi diagram and Dijkstra's algorithm.

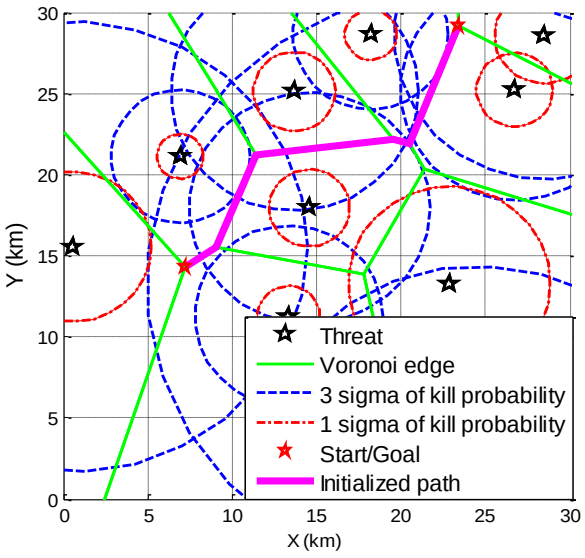

Fig. 11. An initialized path in the navigation space.

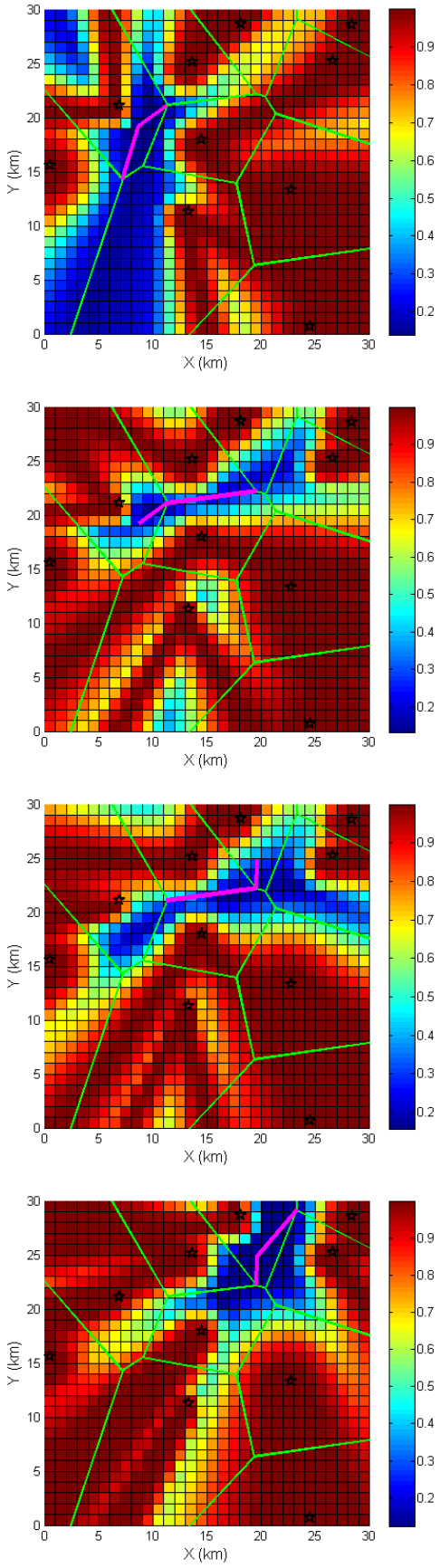

Fig. 12. Distribution of kill probability at each waypoint optimization.

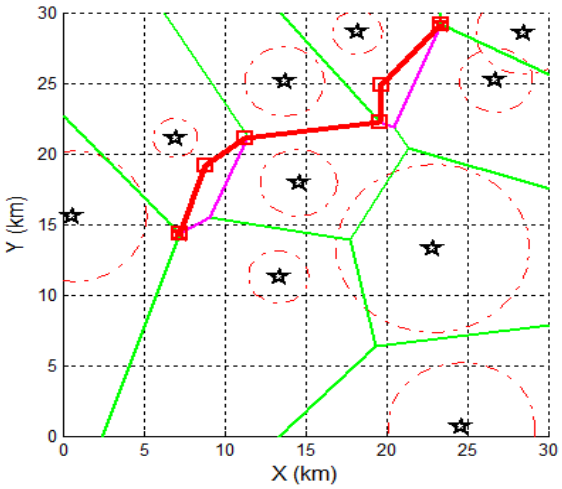

Fig. 13. Resultant optimal path.

Table. 1. Pre/post-optimization changes in survivability

	Before	After
Total	0.5945	0.6871
Seg. 1	0.8848	0.8848
Seg. 2	0.9517	0.9929
Seg. 3	0.8857	0.8928
Seg. 4	0.8928	0.9198
Seg. 5	0.8928	0.9524

Fig. 12 depicts the kill probability calculated at each optimized waypoint shifted from the previous position, with both neighboring waypoints fixed. The figure shows that the kill probability distribution changes according to the initialized path sections, and that the waypoints move toward the lowest kill probability area, thus increasing survivability, which can be verified numerically in Table. 1.

In cases where there are two or more minimum kill probability points, as shown in some diagrams in Fig. 12, a local minimum can be obtained by converging them to the minimum point nearest to the initial position. In this study, the detection time was not integrated into the objective function to prevent the lengthening of the total length of the path, thus decreasing survivability, by deviating far from the initial position when the detection time is considered. Instead, deriving the extreme value closest to the initial position as the optimal solution has the effect of maintaining the path length within an optimal range and presenting results more useful for actual path planning.

2.5 Computation Time

We compared the computation time taken for optimizing two initialized waypoint sets in the same environment by applying the VVCO method and the conventional optimization method, in which all optimization parameters are simultaneously optimized in one run. The number of optimization parameters was varied in the same threat environment, changing only the start and goal points. The threat environment in which the computation time for optimization was compared is as follows: 100 threats in a navigation space of 400 km × 400 km, and the range of standard deviation of the threats was configured to be 10 km $\leq \sigma_i \leq$ 20 km.

In Fig. 14, the blue solid line and red dotted line represent the computation time for optimization when applying the simultaneous optimization method and the VVCO method, respectively. Although the method that optimizes all parameters at the same time shows a steep increase in computation time, as the number of optimization parameters increases, the graph representing the VVCO shows a very small increasing gradient.

2.6 Path Planning Example

We verified the performance of the proposed algorithm with a problem of searching for the optimal path between the given start and goal points, successfully navigating through threats distributed along the path. As shown in Fig. 15, threats (n=100; standard deviation: 5 km$\leq \sigma_i \leq$ 20 km) were generated in a navigation space (400 km × 400 km). The start and goal points are marked with the red star symbol. The solid bold pink line represents the path initialized with the Voronoi diagram and Dijkstra's algorithm. The waypoints constituting this path are used as the baseline values of the waypoint optimization.

Fig. 16 presents the outcome of performing the VVCO. The solid bold red line represents the optimized path. The computation time for optimizing the 24 waypoints initialized with Dijkstra's algorithm was approximately 2.9 s. Table. 2 outlines the optimization levels of survivability before and after the optimization for the entire path and 25 sections. The survivability of the initialized path increased by approximately 10% after applying the optimization algorithm.

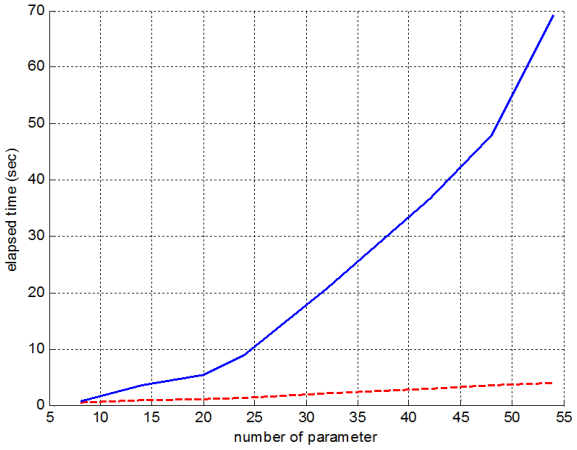

Fig. 14. Comparison of computation time for optimization according to the number of waypoint optimization parameters.

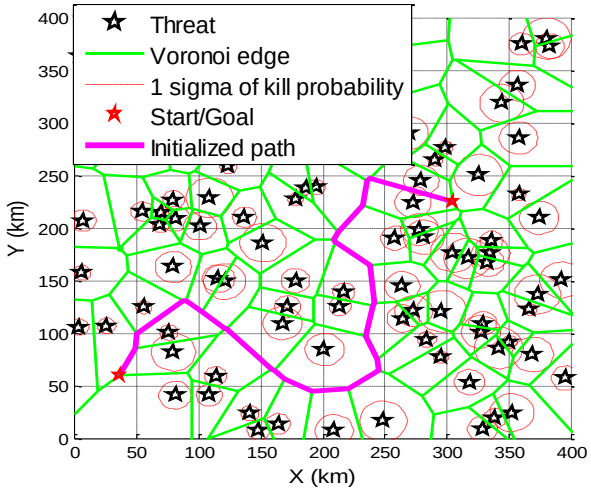

Fig. 15. Environment configuration and initialized path.

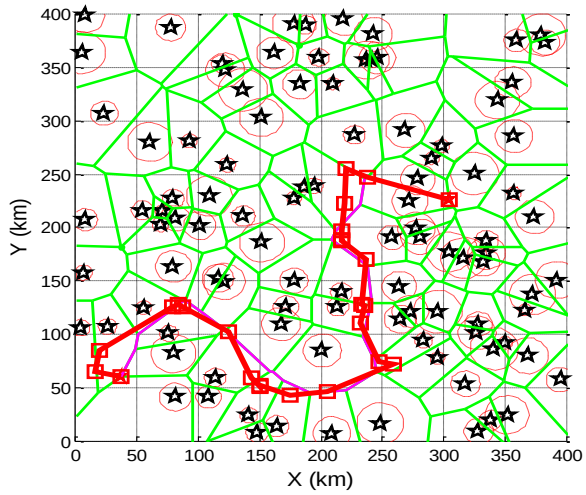

Fig. 16. Optimized path resulting from the VVCO.

Table. 2. Changes in survivability after applying the VVCO

	Before	After
Total	0.3054	0.4041
Seg. 1	0.9524	0.9993
Seg. 2	0.9523	1.0000
Seg. 3	0.9622	0.9827
Seg. 4	0.9896	0.9972
Seg. 5	0.9800	0.9975
Seg. 6	0.9806	0.9941
Seg. 7	0.9989	0.9994
Seg. 8	0.9962	1.0000
Seg. 9	0.9998	1.0000
Seg. 10	0.9995	1.0000
Seg. 11	0.9986	1.0000
Seg. 12	0.9918	0.9982
Seg. 13	0.9911	0.9962
Seg. 14	0.9984	0.9998
Seg. 15	0.9938	0.9975
Seg. 16	0.9821	0.9968
Seg. 17	0.9754	0.9952
Seg. 18	0.9427	0.9850
Seg. 19	0.9958	0.9989
Seg. 20	0.9999	1.0000
Seg. 21	0.9999	1.0000
Seg. 22	0.9996	1.0000
Seg. 23	0.9993	1.0000
Seg. 24	0.9996	0.9996
Seg. 25	0.4233	0.4304

3 Conclusion

In this paper, we proposed a waypoint determination technique that maximizes the survivability of unmanned aerial vehicles (UAVs). A flight path that minimizes the kill probability was calculated using a Voronoi diagram, whereby the threat model was assumed to have a shape of an exponential function of different sizes. However, it is difficult to determine the positions of the waypoints reflecting accurate survival probability with waypoints initialized using a Voronoi diagram. In this study, this problem was solved by optimizing the waypoints initialized with the Voronoi diagram, which yielded route waypoints reflecting the accurate overall survivability kept at the maximum possible level.

Additionally, computation time with a conventional optimization algorithm increases as the navigation space becomes larger and the number of threats increases, because of the increasing number of optimization parameters. Such an increase in computation time makes it difficult to use the optimization algorithm for real-time path planning. In order to solve this problem, this paper proposed a method for optimizing waypoints point by point, and we verified the effectiveness of this method in improving survivability by solving an example problem. The time-saving effect was verified to be increasingly larger as the number of waypoints increases.

A close look at the optimized path may reveal some waypoints to be redundant. This can be adjusted by applying additional logic to the optimized path. For example, distances between waypoints can be narrowed through optimization, and if the distances fall below a certain threshold value, the waypoints involved can be joined by applying relevant logic, or in cases where the routes intersect, only the waypoints in the intersected part can be retained and the other waypoints can be eliminated by applying relevant logic.

The main contribution of this paper is the advantage of the proposed method in initializing waypoints first and then eliminating or adding them with suitable logic, compared with the method of applying any number of waypoints, from one to several hundred, without initializing an adequate number of waypoints. Additionally, it was verified in this study that in an environment where the start and goal points are known, it is more efficient to sequentially optimize parameters in iterations than to simultaneously optimize all parameters in one run.

Acknowledge

This work was supported by the Agency for Defense Development under contract UD150047JD

References

[1] Novy, M. C., Jacques, D. R. and Pachter, M., "Air vehicle optimal trajectories between two radars," American Control Conference, Vol. 1, 2002, pp.785-790.

[2] Hammouri, O. M. and Matalgah, M. M., "Voronoi path planning technique for recovering communication in UAVs," Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp.403-406.

[3] Judd, K. B. and McLain, T. W., "Spline based path planning for unmanned air vehicles," AIAA Guidance, Navigation and Control Conference and Exhibit, Montreal, Canada, 2001.

[4] Neus, M. and Maouche, S., "Motion planning using the modified visibility graph," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 4, 1999, pp. 12-15.

[5] Huang, H. P. and Chung, S. Y., "Dynamic visibility graph for path planning," Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, 2004.

[6] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," International Journal of Robotics Research, Vol. 5, No. 1, 1986, pp. 90-98

[7] Hague, T., Brady, M., and Cameron, S., "Using moments to plan paths for the oxford AGV," IEEE International Conference on Robotics and Automation, Vol. 1, 1990, pp. 210-215.

[8] Connolly, C. I. and Burns, J. B., "Path planning using Laplace's equation," IEEE International Conference on Robotics and Automation, 1990, pp. 2102-2106.

[9] Feder, H. J. S. and Slotine, J. J. E., "Real-time path planning using harmonic potentials in dynamic environments," IEEE International Conference on Robotics and Automation, Vol. 1, 1997, pp. 874-881.

[10] Kabamba, P. T., Meerkov, S. M., and Zeitz, F. H., "Optimal path planning for unmanned combat aerial vehicles to defeat radar tracking," Journal of Guidance, Control, and Dynamics, Vol. 29, No. 2, 2006, pp. 279-288.

[11] Kim, B. S. , Ryoo, C. K., Bang, H., and Chung, E., "Optimal path planning for UAVs under multiple ground threats," Journal of Korean Society for Aeronautical & Space Sciences, Vol. 34, No. 1, 2006, pp. 74-80.

[12] Ruz, J. J., Arevalo, O., de la Cruz, J. M., and Pajares, G., "Using MILP for UAVs trajectory optimization under radar detection risk," Proceedings of 11th IEEE International Conference on Emerging Technologies and Factory Automation, 2006, pp. 1–4.

[13] Besada-Portas, E., de la Torre, L., and de la Cruz, J. M., "Evolutionary trajectory planner for multiple UAVs in realistic scenarios," IEEE Transactions on Robotics, Vol. 26, No. 4, 2010, pp. 619-634.

[14] Lopez, A. S., Zapata, R., and Lama, M. O., "Sampling-based motion planning: a survey," Computaciony Sistemas, Vol. 12, No. 1, 2008, pp. 5–24.

[15] Yang, K., Gan, S. K., and Sukkarieh, S., "An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments," Journal of Intelligent and Robotic Systems, Vol. 57, 2010, pp. 101-122.

[16] Lim, K., Park, S. and Shin, H., "Dynamic construction of the Voronoi diagram for the circle set," Proceedings of the Society of CAD/CAM Conference, 2007.

[17] Mu, L., "Polygon characterization with the multiplicatively weighted Voronoi diagram," The Professional Geographer, Vol. 56, No. 2, 2004, pp.223-239.

[18] Xiao, Q., Gao, X., Fu, X. and Wang, H., "New local path replanning algorithm for unmanned combat air vehicle," Proceedings of the 6th World Congress on Intelligent Control and Automation, 2006, pp.4033-4037.

[19] Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C., Introduction to Algorithms, 2nd, McGraw-Hill Book, 2001.

[20] Dijkstra algorithm, http://adnoctum.tistory.com/165

Copyright Statement

Contact Author Email Address

pkk0725@nate.com
Corresponding author : ckryoo@inha.ac.kr